

(12)特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局(43) 国際公開日
2004 年 4 月 29 日 (29.04.2004)

PCT

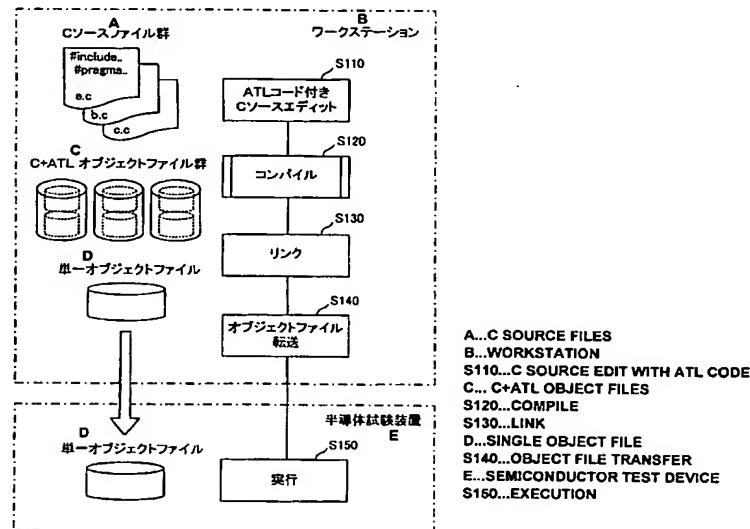
(10) 国際公開番号
WO 2004/036420 A1

- (51) 国際特許分類: G06F 9/44 [JP/JP]; 〒179-0071 東京都練馬区旭町1丁目32番1号 Tokyo (JP).
- (21) 国際出願番号: PCT/JP2003/013302
- (22) 国際出願日: 2003 年 10 月 17 日 (17.10.2003)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (30) 優先権データ:
特願 2002-305010
2002 年 10 月 18 日 (18.10.2002) JP
- (71) 出願人 (米国を除く全ての指定国について): 株式会社アドバンテスト (ADVANTEST CORPORATION)
- (72) 発明者: 前田 裕紀 (MAEDA, Hironori) [JP/JP]; 〒179-0071 東京都練馬区旭町1丁目32番1号 株式会社アドバンテスト内 Tokyo (JP).
- (74) 代理人: 酒井 宏明 (SAKAI, Hiroaki); 〒100-0013 東京都千代田区霞が関三丁目2番6号 東京倶楽部ビルディング Tokyo (JP).
- (81) 指定国 (国内): DE, KR, US.
- 添付公開書類:
— 国際調査報告書

[続葉有]

(54) Title: PROGRAM DEVELOPMENT SUPPORT DEVICE, PROGRAM EXECUTION DEVICE, COMPILE METHOD AND DEBUG METHOD

(54) 発明の名称: プログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法



(57) Abstract: On a program development device (such as a workstation), created is a heterogeneous language source program where a description of a proprietary language source program (such as in the ATL) is embedded in a preprocessor describing section of a general purpose language source program (such as in the C language). From the heterogeneous language source program, the general purpose source program describing section and the particular specification language source program describing section are extracted, and respectively compiled by their compilers to obtain their object codes. By combining the object codes a single object file is created.

(57) 要約: プログラム開発装置 (例えば、ワークステーション) 上で、汎用言語ソースプログラム (例えば、C 言語) のプリプロセッサ記述部に、独自仕様言語ソースプログラム (例えば、ATL) の記述を埋め込んだ異種言語混在ソースプログラムを作成する。そして、その異種言語混在ソースプログラムから汎用言語ソースプログラム記述部と独自仕様言語ソースプログラム記述部を取り出して、それぞれのコンパイラでコンパイルし、得られたそれぞれのオブジェクトコードを

[続葉有]



2文字コード及び他の略語については、定期発行される各PCTガゼットの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明 細 書

プログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法

5

技術分野

本発明は、半導体試験装置等の被制御装置に対する制御コマンドが記述された独自仕様言語プログラムと、独自仕様言語プログラムの実行ステップや独自仕様言語プログラムから得られたデータの処理ステップ等が記述された汎用言語プログラムとを開発するためのプログラム開発支援装置、それらプログラムを実行するプログラム実行装置、それらプログラムのコンパイル方法およびデバッグ方法に関し、特に、上記した独自仕様言語プログラムと汎用言語プログラムとが一つのファイル内に混在して記述された異種言語混在プログラムを開発するためのプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法に関する。

15

背景技術

膨大かつ多様な信号処理が求められる測定装置や通信装置などの電子機器の多くには、高性能なプロセッサが搭載されている。そのようなプロセッサ上で実行されるプログラム（ファームウェア）もまた、複雑でサイズが大きくなる傾向にある。必然と、これら電子機器に記録されているプログラムは、未知のバグを含んでいたり、機能の追加や改善を求められることが多い。

20

そのため、このような電子機器は、動作の設定／監視や上記したプログラムのアップデートを可能にするために、通信ケーブルを介して、外部のコンピュータと接続可能であることが多い。すなわち、外部のコンピュータから電子機器のプロセッサに対し、制御コマンドや制御プログラム自体を配信することが可能である。さらに換言すれば、外部のコンピュータ上で開発したプログラムのアルゴリ

25

ズムや設定内容に従って、電子機器を操作することが可能である。

そのような電子機器のうちでも特に半導体試験装置は、多種多様かつ独自仕様の半導体デバイスに対してそれぞれ個別のデバイステストプログラムを用意する必要がある特殊な測定装置である。特に、半導体試験装置のプロセッサ上で実行
5 させるプログラムは、利用者（半導体デバイス製造メーカ）自らによって開発されるという形態が定着している。

ところが、半導体試験装置を筆頭とした特殊用途の電子機器は、搭載されるプロセッサも独自仕様のものが多く、必然とそのプロセッサが理解できるバイナリ
10 ファイルも独自仕様に従っている。そのため、そのようなバイナリファイルの元となるソースファイルの作成に必要なプログラム言語の仕様や開発支援環境もまた特殊となり、プログラム開発者は、プログラム言語とともに開発支援環境の操作も習熟する必要があった。

このような背景から、近年においては、C言語やJ A V A（登録商標）等の汎
15 用言語によって開発されたプログラムを実行することができる電子機器も登場してきている。ところが、そのような電子機器を採用することは、過去の独自仕様のプログラム資産を活用することができないという新たな問題を生み出す。

この問題を解決するために、データ処理やアルゴリズム全体が、C言語のような汎用言語プログラムで記述され、その汎用言語プログラム内から、電子機器に
20 依存する独自仕様言語プログラムがサブルーチンとして呼び出されるという手法がある。以下に、この手法に従ったプログラムの開発と実行に関し、プログラムを実行する電子機器が半導体試験装置である場合を例に挙げて詳述する。

まず、半導体試験装置上でのプログラムの実行動作を理解するために、半導体
試験装置とそのプログラム開発環境が説明される。半導体試験装置は、半導体メモ
25 リ、ロジック I C、リニア I Cなどの半導体デバイスに対して所定の動作試験を行う特殊な測定装置であり、一般にその装置構成は、上記した半導体デバイスの種別ごとに異なる。また、半導体試験装置への試験実行指示、試験結果取得およびプログラム開発は、通常、半導体試験装置に接続されたワークステーション

によって行なわれる。半導体試験は、例えば、特開 2001-195275 号公報に開示されているように、半導体試験装置とワークステーションとで構成される半導体試験システムによって実現される。

第 10 図は、従来の半導体試験システムの概略構成を示すブロック図であり、
5 特に、上記したように異なる装置構成の半導体試験装置間において共通する構成を示すものである。第 10 図において、半導体試験装置 100 は、テストプロセッサ 110 と、テスト本体 120 と、テストヘッド 130 と、通信インターフェース 140 とを備えて構成される。

テストプロセッサ 110 は、テスト本体 120 との間で、制御コマンドの送信
10 や試験データの送受信を行う手段であり、テスト本体 120 の制御や後述するワークステーションとの間の通信を行うコントローラである。特に、テストプロセッサ 110 は、内部に搭載したメモリ（図示略）に、OS（オペレーティングシステム）カーネル 111 を格納しており、デバイステストプログラムの起動や監視、メモリ管理を行うとともに、同じくメモリに格納された通信バスドライバ 1
15 12 やテストバスドライバ 113 を介して、通信インターフェース 140 の監視／制御、テスト本体 120 の制御、試験データの送受信を行う。

デバイステストプログラムは、上記した汎用言語プログラム 114 と独自仕様
言語プログラム 117 とで構成されており、それら全体で、被測定デバイス 13
1 に対する機能試験や DC パラメトリック試験等の各種の試験を実施する手順を
20 規定している。汎用言語プログラム 114 は、試験結果として取得した各種データを処理するコマンドを含んだステートメントと、デバイステストプログラム全体をどのように実行するかを示すコマンドを含んだステートメントとによって構成されており、OS カーネル 111 上で直接実行可能なバイナリファイルである。

一方、独自仕様言語プログラム 117 は、テスト本体 120 を制御するための
25 コマンドで構成されるオブジェクトファイルである。そのオブジェクトファイルは、過去の資産である独自仕様言語プログラムと同様に、独自仕様言語プログラム 117 に最適化されたカーネル上でのみ直接実行可能なバイナリファイルであ

る。そのため、独自仕様言語プログラム 117 を OS カーネル 111 上で実行させるにあたっては、実行用エミュレータ 115 による解釈処理が要求される。また、独自仕様言語プログラム 117 は、後述するワークステーション 200 に対してのディスク・アクセス、キー入力、ディスプレイ表示といった入出力コマンドも含んでおり、それら入出力コマンドの実行にあたっては、実行用エミュレータ 115 による解釈に加え、さらに I/O 制御用エミュレータ 116 による解釈処理が要求される。

テスト本体 120 は、テストプロセッサ 110 から送信される制御コマンドに従って、テストヘッド 130 に実装された被測定デバイス 131 に対し、機能試験や DC パラメトリック試験、RF 試験（高周波試験）等の各種の試験を行う手段であり、レジスタ 121、メモリ 122、試験信号送受信部 123 を備えて構成される。レジスタ 121 は、テストプロセッサ 110 内のテストバスドライバ 113 との間で送受信される各種のデータを格納し、格納されたデータは、直接またはメモリ 122 を介して試験信号送受信部 123 に送信される。

また、試験信号送受信部 123 から出力されるデータは、一旦レジスタ 121 やメモリ 122 に格納された後、レジスタ 121 を介してテストプロセッサ 110 内のテストバスドライバ 113 に送信される。試験信号送受信部 123 は、パターン発生器やタイミング発生器、DC ユニット等の種々の試験ユニットから構成され、これら試験ユニットによって生成された試験信号を被測定デバイス 131 に入力するとともに、被測定デバイス 131 の出力ピンに現れるデータを取得する。

第 11 図は、上記したワークステーション 200 の概略構成を示すブロック図である。ワークステーション 200 は、半導体試験装置 100 内のテストプロセッサ 110 に対してプログラムの転送や実行指示を行うコンソール端末の役割と、汎用言語プログラム 114 や独自仕様言語プログラム 117 の開発を支援するプログラム開発支援装置の役割とを担う。第 11 図において、ワークステーション 200 は、プロセッサ 220 と、通信インターフェース 241 と、ハードディスク

ク装置 242 と、マウス 243 と、キーボード 244 と、ディスプレイ装置 245 とを備えて構成される。

プロセッサ 220 は、内部に搭載したメモリ（図示略）に、OS（オペレーティングシステム）カーネル 221 を格納しており、種々のプログラムの起動や監視、メモリ管理を行うとともに、同じくメモリに格納される通信バスドライバ 223、ハードディスクドライバ 224、マウสดライバ 225、キーボードドライバ 226、ディスプレイドライバ 227 を介して、通信インターフェース 241 の監視や制御、ハードディスク装置 242 との間でのプログラムやデータの読み込み／書き込み、マウス 243 やキーボード 244 からの入力情報の取得、ディスプレイ装置 245 への表示情報の出力を行う。ここで特に、通信インターフェース 241 は、通信ケーブル（図示せず）を介して、第 10 図に示した通信インターフェース 140 と接続されており、ワークステーション 200 と半導体試験装置 100 との間の通信を可能にさせる。

また、OSカーネル 221 は、GUI（Graphical User Interface）処理部 222 を有する。図示されるエディタ 228、汎用言語コンパイラ 229、リンカ 233、汎用言語デバッガ 231、独自仕様言語コンパイラ 230、独自仕様言語デバッガ 232 などの各種プログラムは、ディスプレイ装置 245 上に表示されるウィンドウ画面単位で実行されることが可能である。このワークステーション 200 は、汎用的なコンピュータの装置構成と同等である。よって、上記した各種ドライバや各種プログラムは、通常はハードディスク装置 242 に記憶されており、OSカーネル 221 に従って必要に応じて上記メモリに読み込まれて実行される。

次に、半導体試験装置 100 とワークステーション 200 とで構成される半導体試験システムにおいて、デバイステストプログラムの開発と実行の流れが説明される。第 12 図は、従来のデバイステストプログラムの開発と実行手順を示したフローチャートである。ここでは、デバイステストプログラムが、上記したように汎用言語プログラムと独自仕様言語プログラムとで構成されており、汎用言

語プログラムとしてC言語を採用し、独自仕様言語プログラムとしてATL（アドバンテスト社独自規格）を採用した場合を例に挙げる。

まず、プログラム開発者は、ワークステーション200上においてエディタ228を起動させ、C言語によるソースプログラムを作成する（ステップS301）。このソースプログラムは、上述したように、デバイステストプログラム全体のアルゴリズムを記述しており、シーケンス処理の所望の位置で、ATLで記述されたオブジェクトプログラムを呼び出して実行したり、その実行によって得られた試験結果データを処理する手順を規定する。

C言語によるソースプログラムの作成を終えると、プログラム開発者は、Cコンパイラ（汎用言語コンパイラ229に相当）に対し、作成したソースプログラムのファイル（以下、必要なヘッダファイル等を含めてCソースファイルと称する。）を指定してコンパイルを実行させる（ステップS302）。コンパイル処理では、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ228でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合には、上記したCソースファイルを機械語に翻訳したオブジェクトファイル（以下、Cオブジェクトファイルと称する。）が生成される。

ステップS301で作成された複数のCソースファイルに対し、ステップS302が完了すると、プログラム開発者は、リンカ233に対し、生成された複数のCオブジェクトファイルやその他に必要なライブラリファイルを指定してリンクを実行させる（ステップS303）。このリンクによって、半導体試験装置100のテストプロセッサ110上で直接実行可能な単一のCオブジェクトファイルが生成される。

また、プログラム開発者は、C言語によるオブジェクトファイルの生成と並行して、ワークステーション200上においてエディタ228を起動させ、ATLによるソースプログラムを作成する（ステップS401）。このソースプログラムは、上述したように、半導体試験装置100を制御するための制御コマンドを

記述する。

ATLによるソースプログラムの作成が終えると、プログラム開発者は、ATLコンパイラ（独自仕様言語コンパイラ230に相当）に対し、作成したソースプログラムのファイル（以下、ATLソースファイルと称する。）を指定してコンパイルを実行させる（ステップS402）。なお、このコンパイル処理でも、
5 上記したステップS302と同様に、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ228でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合には、上記したATLソースプログラムは、上記したCオブジェクトファイルで表わされる機械語（ある特定の
10 テスタプロセッサで理解できる機械語）とは異なる旧テスタプロセッサ仕様の機械語に翻訳され、オブジェクトファイル（以下、ATLオブジェクトファイルと称する。）が生成される。

このような手順によって、単一CオブジェクトファイルとATLオブジェクトファイル群が用意されると、プログラム開発者は、ワークステーション200上
15 において、半導体試験装置100との通信を可能にするコントロールプログラムを起動し、そのコントロールプログラムを用いて単一CオブジェクトファイルとATLオブジェクトファイル群を半導体試験装置100のテスタプロセッサ110へと転送する（ステップS304、ステップS403）。

続いて、プログラム開発者は、上記したコントロールプログラムに対して、単
20 一Cオブジェクトファイルの実行指示を与える（ステップS305）。これにより、半導体試験装置100のテスタプロセッサ110は、単一Cオブジェクトファイルに記述されたアルゴリズムに従って、ATLオブジェクトファイルの実行→テスタ本体120の所望の試験ユニットの稼動→被測定デバイス131から得られた試験結果の取得→データ処理を繰り返す。この際、データ処理によって適
25 宜加工等が行われた試験結果は、半導体試験装置100の通信インターフェース140、通信ケーブル、ワークステーション200の通信インターフェース241を介して、上記したコントロールプログラムで受け取ることができ、そのコン

トロールプログラムに割り当てられたウィンドウ画面上に表示される。

ここで、プログラム開発者は、試験結果が明らかに異常である場合等の不具合を発見した場合、デバイステストプログラムに論理的なエラーが含まれていると判断し、ワークステーション 200 上において汎用言語デバッガ 231 を起動させ、Cソースファイル中の所定のステートメントにブレークポイントを設定する。そして、プログラム開発者がデバッグ開始を指示することにより、汎用言語デバッガ 231 は、再度、上記したステップ S302～S305 の手順によって単一 C オブジェクトファイルを実行し、実行されたステートメントが、設定したブレークポイントに達したことを検出すると、ブレークしたステートメントの段階で有効となっている変数を表示する。プログラム開発者は、この変数の確認によって論理的なエラーを発見すると、エディタ 228 を起動させ、Cソースファイルを適宜修正して上記したステップ S302～S305 の手順を繰り返す。

一方、プログラム開発者は、汎用言語デバッガ 231 によって C ソースファイルの論理的なエラーを発見できないときは、続いて、独自仕様言語デバッガ 232 を起動させ、ATL ソースファイル中の所定のステートメントにブレークポイントを設定し、上記同様のデバッグ処理を行う。

しかしながら、上記したように、被制御装置（上記例では半導体試験装置）上で実行させるプログラムが汎用言語と独自仕様言語のように異種言語で記述される場合には、独自仕様言語による過去のプログラム資産を活用することができるものの、双方のプログラムの開発段階において、それぞれ個別のソースファイルが必要とされる。上記した例では、C ソースファイルと ATL ソースファイルとをそれぞれ別ファイルとして用意する必要がある。簡単に言えば、汎用言語で記述されたソースファイル中に独自仕様言語に作成されたオブジェクトファイルの呼び出しを埋め込んだ場合には、少なくとも 2 つのソースファイルが必要となる。特に、ある一つの汎用言語ソースファイルに対しては、ある特定の独自仕様言語ソースファイルが対応するため、これらファイルはひとまとめで管理しなければならない。

さらに、双方のソースファイルの内容から、関連する異種言語のソースファイルを特定することは困難であり、一度管理を誤ると、対応関係を再度見出すのに多くの時間と手間を要するという問題があった。このことから、エディタ上でのソースファイルの修正作業や他のソースファイルの部分的な流用については慎重にならざるを得ず、上記問題は、プログラム開発者にとってもミスを増やす原因となっていた。

また、一つの実行プログラムに対して、汎用言語ソースファイルと独自仕様言語ソースファイルとを用意することから、必然と、それらをコンパイルすることによって同じ数だけオブジェクトファイルが生成される。これは、上記した管理がさらに複雑になることを意味する。このように、従来では、一つの実行プログラムに対して異種言語の複数のソースファイルおよびオブジェクトファイルが存在したため、ファイル管理が煩雑となり、プログラムの開発効率を低下させるという問題があった。

本発明は上記に鑑みてなされたものであって、汎用言語ソースファイルのプリプロセッサ記述部に独自仕様言語のソースファイルを埋め込むことによって、独自仕様言語のプログラムによる過去の資産を活用できるとともに、一つの実行ファイルに対して必要なソースファイルおよびオブジェクトファイルの数を大幅に減少させることのできるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法を提供することを目的とする。

発明の開示

上記目的を達成するために、本発明にかかるプログラム開発支援装置は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのプログラム開発支援装置において、前記独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイル手段（後述する独自仕様言語コンパイラ

30に相当)と、前記異種言語混在ソースプログラム内の前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイル手段(後述する汎用言語コンパイラ29に相当)と、前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出し、抽出した独自仕様言語ソースプログラムを指定して前記独自仕様言語コンパイル手段を実行させるとともに、前記異種言語混在ソースプログラムを指定して前記汎用言語コンパイル手段を実行させ、得られた独自仕様言語オブジェクトコードと汎用言語オブジェクトコードを結合してオブジェクトファイルを生成する統合コンパイル手段(後述する統合コンパイラ34に相当)と、前記統合コンパイル手段によって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイル

5 10 15 20 25

を生成するリンク手段(後述するリンカ33に相当)と、を備えたことを特徴としている。

また、本発明にかかるプログラム実行装置は、汎用言語ソースプログラムのオブジェクトコードと独自仕様言語ソースプログラムのオブジェクトコードが混在したプログラムファイルを実行するプログラム実行装置(後述する半導体試験装置11に相当)において、前記プログラムファイルの実行開始時に、前記汎用言語ソースプログラムのオブジェクトコードと前記独自仕様言語ソースプログラムのオブジェクトコードをメモリにロードすることを特徴としている。

また、本発明にかかるコンパイル方法は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのコンパイル方法において、前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出する独自仕様言語ソースプログラム抽出ステップ(後述するステップS121に相当)と、抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイルステップ(後述するステップS123に相当)と、前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分を

コンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップ（後述するステップS 1 2 2に相当）と、前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップ（後述するステップS 1 2 4に相当）と、

- 5 前記オブジェクトファイル生成ステップによって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンクステップ（後述するステップS 1 3 0に相当）と、を含んだことを特徴としている。

- また、本発明にかかるデバッグ方法は、汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム実行装置上で実行可能なプログラムファイルをデバッグするためのデバッグ方法において、前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定ステップと、前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し（後述するステップS 2 0 3に相当）、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動する（後述するステップS 2 0 6に相当）デバッガ起動ステップと、前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報（後述するステップS 2 0 4，ステップS 2 0 7に相当）を共通するウィンドウ画面に表示するデバッグ情報表示ステップ（後述するステップS 2 0 5に相当）と、を含んだことを特徴としている。

また、本発明にかかるコンピュータ読み取り可能な記録媒体は、上記コンパイル方法をコンピュータに実行させることを特徴としている。

- 25 また、本発明にかかるコンピュータ読み取り可能な記録媒体は、上記デバッグ方法をコンピュータに実行させることを特徴としている。

図面の簡単な説明

第1図は、実施の形態にかかる半導体試験システムの概略構成を示すブロック図であり、第2図は、デバイステストプログラムの開発と実行手順を示したフローチャートであり、第3図は、C＋A T Lソースプログラムの記述例を示す図であり、第4図は、統合コンパイラによるコンパイル処理を説明するためのフローチャートであり、第5図は、A T Lソースファイルの生成処理を説明するための説明図であり、第6図は、C＋A T Lオブジェクトファイルの構成を示す図であり、第7図は、デバッガ選択ルーチンを示すフローチャートであり、第8図は、A T L記述部内でブレークした状態の統合デバッガの実行画面の例を示す図であり、第9図は、C言語記述部内でブレークした状態の統合デバッガの実行画面の例を示す図であり、第10図は、従来の半導体試験システムの概略構成を示すブロック図であり、第11図は、従来の半導体試験システムのワークステーションの概略構成を示すブロック図であり、第12図は、従来のデバイステストプログラムの開発と実行手順を示したフローチャートである。

発明を実施するための最良の形態

以下に、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法の実施の形態を図面に基づいて詳細に説明する。この実施の形態により本発明は限定されない。

この実施の形態では、本発明の特徴の理解を容易にするために、上述した従来技術の説明と同様に、本発明を半導体試験装置とワークステーションとから構成される半導体試験システムに適用した場合の形態が説明される。具体的には、本発明にかかるプログラム開発支援装置が、半導体試験システムのワークステーションに相当し、本発明にかかるプログラム実行装置が半導体試験システムの半導体試験装置に相当し、本発明にかかるコンパイル方法およびデバッグ方法が上記半導体試験システム上でのコンパイル方法およびデバッグ方法に相当する。

第1図は、本実施の形態にかかる半導体試験システムを示すブロック図である。

第1図に示す半導体試験システムは、通信ケーブルで接続されたワークステーション10と半導体試験装置11とを備えて構成される。

ワークステーション10の基本構成は、第11図に示された従来の半導体試験システムのワークステーション200と同じであり、第1図中のプロセッサ20と、通信インターフェース41と、ハードディスク装置42と、マウス43と、キーボード44と、ディスプレイ装置45は、それぞれ第11図に示したプロセッサ220、通信インターフェース241、ハードディスク装置242、マウス243、キーボード244、ディスプレイ装置245にそれぞれ対応する。

また、プロセッサ20内部において、メモリ（図示略）に格納されるOSカーネル21、GUI処理部22、通信バスドライバ23、ハードディスクドライバ24、マウスドライバ25、キーボードドライバ26、ディスプレイドライバ27、エディタ28、汎用言語コンパイラ29、独自仕様言語コンパイラ30、汎用言語デバッガ31、独自仕様言語デバッガ32、リンカ33もまた、それぞれ第11図に示したOSカーネル221、GUI処理部222、通信バスドライバ223、ハードディスクドライバ224、マウスドライバ225、キーボードドライバ226、ディスプレイドライバ227、エディタ228、汎用言語コンパイラ229、独自仕様言語コンパイラ230、汎用言語デバッガ231、独自仕様言語デバッガ232、リンカ233にそれぞれ対応する。

本実施の形態で説明するワークステーション10は、汎用言語コンパイラ29と独自仕様言語コンパイラ30の上位アプリケーションに位置する統合コンパイラ34を有している点で、従来のワークステーション200と異なる。換言すれば、このワークステーション10は、統合コンパイラ34から、汎用言語コンパイラ29と独自仕様言語コンパイラ30をそれぞれ利用することができる。

さらに、本実施の形態で説明するワークステーション10は、汎用言語デバッガ31と独自仕様言語デバッガ32の上位アプリケーションに位置する統合デバッガ35を有している点で、従来のワークステーション200と異なる。すなわち、統合コンパイラ34と同様に、このワークステーション10は、統合デバ

ガ 3 5 から、汎用言語デバッガ 3 1 と独自仕様言語デバッガ 3 2 をそれぞれ利用することができる。

また、第 1 図に示す半導体試験システムでは、半導体試験装置 1 1 はワークステーション 1 0 に接続されているが、この半導体試験装置 1 1 の内部構成は第 1 0 図に示した従来の半導体試験装置 1 0 0 と同様である。しかし、この半導体試験装置 1 1 は、ワークステーション 1 0 において開発されるプログラムの態様に依存して、テストプロセッサの動作が従来と一部異なる。

次に、この半導体試験システムにおいて、デバイステストプログラムの開発と実行の流れが説明される。第 2 図は、本実施の形態におけるデバイステストプログラムの開発と実行手順を示したフローチャートである。ここでは、第 1 2 図の説明と同様に、デバイステストプログラムが、汎用言語プログラムと独自仕様言語プログラムとで構成されており、汎用言語プログラムとして C 言語を採用し、独自仕様言語プログラムとして A T L (アドバンテスト社独自規格)を採用した場合を例に挙げる。

まず、プログラム開発者は、ワークステーション 1 0 上においてエディタ 2 8 を起動させ、ソースプログラムを作成する (ステップ S 1 1 0)。このソースプログラムは、汎用言語である C 言語で記述されるが、第 1 2 図で説明した C ソースファイルの内容とは異なり、プリプロセッサ記述部に、独自仕様言語である A T L ソースファイルの内容を記述する。このソースプログラムを C + A T L ソースプログラムと称する。

第 3 図は、C + A T L ソースプログラムの記述例を示す図である。なお、第 3 図において、左端に配置された「数字 :」は、説明の便宜のために用いられる行番号を示すが、実際のプログラム動作においては無視される。以下の C + A T L ソースプログラムの内容の説明では、この行番号によって各ステートメントが参照される。

C 言語コンパイラは、先頭が # で続くステートメントをプリプロセッサコマンドであると認識する。第 3 図に示す C + A T L ソースプログラムでは、行番号 1

の `#include`、行番号 3, 4, 5 の `#pragma` がそのプリプロセッサコマンドに相当する。`#include` は、ヘッダファイル「AT/hybrid.h」の記述内容を単にその位置に展開するコマンドであり、行番号 10 以降に続くメイン関数内において必要となる内容である。

5 一方、`#pragma` は、C 言語との全般的な互換性を保ちながら、マシンまたは OS に固有の機能を実現する特別なプリプロセッサコマンドである。よって、`#pragma` は定義上、マシンまたは OS に固有であり、通常、コンパイラごとに異なる。`#pragma` は、本来、プリプロセッサに新しい機能を与えたり、インプリメントに依存する情報をコンパイラに与えるために利用されるが、本実施
10 形態で作成される C+ATL ソースプログラムでは、`#pragma` によって渡されるトークンとして、独自仕様言語である ATL の記述を埋め込む。この `#pragma` によって渡される ATL の記述の処理は後述される。

第 3 図に示す C+ATL ソースプログラムにおいて、行番号 10 ~ 28 に記述されている内容は、従来の半導体試験システムのワークステーション上において
15 作成される内容と同じであり、その記述の中において、ATL オブジェクトファイルの呼び出しやデータ処理が規定される。

C+ATL ソースプログラムの作成が終了すると、プログラム開発者は、統合コンパイラ 34 に対し、作成した C+ATL ソースプログラムのファイル（以下、必要なヘッダファイル等を含めて C+ATL ソースファイルと称する。）を指定
20 してコンパイルを実行させる（ステップ S120）。このコンパイルの実行に際しては、まず構文チェックが行なわれ、構文エラーが生じた場合には、プログラム開発者は、エディタ 28 でエラー箇所を修正し、再度コンパイルの実行を指示する。エラーがない場合に初めて、オブジェクトファイルを生成するためのコンパイル処理が開始される。

25 第 4 図は、統合コンパイラ 34 によるコンパイル処理を説明するためのフローチャートである。統合コンパイラ 34 は、ステップ S110 において作成された C+ATL ソースファイルにおいて構文エラーを発見できなかったときは、続き

て、そのC+ATLソースファイルからATL記述部を抽出し、ATLソースファイルを生成する（ステップS121）。第5図は、このATLソースファイルの生成処理を説明するための説明図である。ATLソースファイルの生成処理は、上述したように、C+ATLソースファイルのプリプロセッサ記述部から、# p
 5 r a g m a を識別し、# p r a g m a の後に続くトークンを解析することから始まる。第5図に示す例では、# p r a g m a の直後の a t l が、その後に続く情報がATL記述部であることを示すキーワードとなる。

第5図の例を用いて具体的な説明を行うと、統合コンパイラ34は、行番号3において、# p r a g m a a t l を認識すると、その後に続くキーワードの n
 10 a m e を取り出し、そのキーワード n a m e に続くダブルクォーテーションで囲まれた記述、すなわち S A M P L E がプログラム名であると解釈する。この解釈によって、ATLソースファイルの先頭部に、P R O S A M P L E が挿入される。つぎに、統合コンパイラ34は、行番号4において、# p r a g m a a t
 15 l を認識すると、その後に続くキーワードの s o c k e t を取り出し、そのキーワード s o c k e t に続くダブルクォーテーションで囲まれた記述、すなわち S S O C が使用するATLのソケットプログラム名であると解釈する。この解釈によって、ATLソースファイル内の P R O S A M P L E の後に S S O C が挿入される。

さらに、統合コンパイラ34は、行番号5において、# p r a g m a a t l
 20 を認識すると、その後に続くキーワードの p r o c を取り出し、そのキーワード p r o c の直後の文字列とその後に続くダブルクォーテーションで囲まれた記述、すなわち、

```
P1 (ARG1, ARG2 (2)) " {WRITE " ARG1=" , ARG1, /WRITE " ARG2=" , ARG2, /} "
```

25 が関数定義であると解釈する。この解釈によって、ATLソースファイルに、

```
P1 : ARGUMENT (ARG1, ARG2 (2))  
      WRITE " ARG1=" , ARG1; /
```

```
WRITE " ARG 2 =" , ARG 2, /  
GOTO CONTINUE
```

が追記される。

そして、統合コンパイラ 34 は、C+ATL ソースファイルにおいて、# p r
5 a g m a a t l を発見することができないまま、C+ATL ソースファイルの
最終行（行番号 28）に達すると、ATL ソースファイルの最後に END を挿入
してその ATL ソースファイルの作成を終える。

ATL ソースファイルの作成が終わると、統合コンパイラ 34 は、C+ATL
ソースファイルの C 言語記述部のコンパイル、すなわち C オブジェクトコードの
10 生成を行う（ステップ S 122）。このコンパイルは、通常の C コンパイラ（汎
用言語コンパイラ 29 に相当する。）によって処理され、上記した # p r a g m
a の記述部分は無視される。統合コンパイラ 34 が C コンパイラを呼び出して処
理させるとしているが、統合コンパイラ 34 自体に C コンパイラの機能を取り入
れ、上記した ATL ソースファイルの生成と並行して、C オブジェクトコードの
15 生成が行われてもよい。この場合、第 1 図に示した汎用言語コンパイラ 29 は不
要となる。

C オブジェクトコードの生成が終わると、統合コンパイラ 34 は、ステップ S
121 において生成された ATL ソースファイルのコンパイル、すなわち ATL
オブジェクトコードの生成を行う（ステップ S 123）。このコンパイルは、A
20 TL コンパイラ（独自仕様言語コンパイラ 30 に相当する。）の呼び出しによっ
て実行され、第 12 図のステップ S 402 と同様に、上記した C オブジェクトコ
ードで表わされる機械語（ある特定のテストプロセッサで理解できる機械語）と
は異なる旧テストプロセッサ仕様の機械語への翻訳を行う。

ATL オブジェクトコードの生成が終わると、統合コンパイラ 34 は、ステッ
25 プ S 122 において生成された C オブジェクトコードに、ステップ S 121 にお
いて生成された ATL オブジェクトコードを結合し、さらにその ATL オブジェ
クトコードが格納された位置情報（ATL オブジェクトコード開始位置）を付加

したオブジェクトファイル（以下、C＋ATLオブジェクトファイルと称する。）を生成する（ステップS124）。第6図は、このC＋ATLオブジェクトファイルの構成を示す図である。第6図に示すように、C＋ATLオブジェクトファイルは、Cオブジェクトコードに続いてATLオブジェクトコードが配置される。同図において、ATLオブジェクトコードの位置情報等の付加情報の図示は省略されている。

この統合コンパイラ34によるコンパイル処理は、上記同様に作成された複数のC＋ATLソースファイルに対して行われ、これにより複数のC＋ATLオブジェクトファイルが用意される。このようにして統合コンパイラ34によるコンパイル処理が終わると、プログラム開発者は、リンカ33に対し、生成された複数のC＋ATLオブジェクトファイルやその他に必要なライブラリファイルを指定してリンクを実行させる（第2図のステップS130）。

リンカ33は、複数のC＋ATLオブジェクトファイルやその他に必要なライブラリファイルに加え、上記各C＋ATLオブジェクトファイルからATLオブジェクトコード部分をロードするためのロードプログラムを用意し、それらをリンクすることによって、半導体試験装置11のテストプロセッサ上で直接実行可能な単一オブジェクトファイルを生成する。

このような手順によって、単一オブジェクトファイルが用意されると、プログラム開発者は、ワークステーション10上において、半導体試験装置11との通信を可能にするコントロールプログラムを起動し、そのコントロールプログラムを用いて上記単一オブジェクトファイルを半導体試験装置11のテストプロセッサへと転送する（ステップS140）。

続いて、プログラム開発者は、上記したコントロールプログラムに対して、単一オブジェクトファイルの実行指示を与える（ステップS150）。これにより、半導体試験装置11のテストプロセッサは、まず、単一オブジェクトファイルに含まれているロードプログラムに従って、同単一オブジェクトファイル内に配置されているCオブジェクトコードとATLオブジェクトコードをメモリ上にロー

ドする。続いて、テストプロセッサは、ロードされたCオブジェクトコードに記述されたアルゴリズムに従って、同じくロードされているATLオブジェクトコードの実行→テスト本体の所望の試験ユニットの稼動→被測定デバイスから得られた試験結果の取得→データ処理を繰り返す。この際、データ処理によって適宜加工等が行われた試験結果は、従来と同様に、半導体試験装置11の通信インターフェース、通信ケーブル、ワークステーション10の通信インターフェース41を介して、上記したコントロールプログラムで受け取ることができ、そのコントロールプログラムに割り当てられたウィンドウ画面上に表示される。

なお、ここでは、単一オブジェクトファイル内に、ロードプログラムが含まれているとしたが、半導体試験装置11のテストプロセッサ内において、あらかじめそのロードプログラムが読み込まれ、ワークステーション10からの実行指示に従って、最初にこのロードプログラムが起動されてもよい。

次に、本実施の形態における半導体試験システムのデバック処理が説明される。プログラム開発者は、ステップS150の実行によって得た試験結果が、明らかに異常である場合等の不具合を発見した場合、従来と同様、デバイステストプログラムに対してデバック処理を行う。まず、プログラム開発者は、ワークステーション10上において、統合デバッガ35を起動させ、C+ATLソースファイル中の所定のステートメントにブレークポイントを設定する。

そして、プログラム開発者がデバッグ開始を指示することによって、統合デバッガ35は、再度、上記したステップS120～S150の手順によって単一オブジェクトファイルを実行し、実行されたステートメントが、設定したブレークポイントに達したことを検出すると、Cデバッガ（汎用言語デバッガ31に相当する。）とATLデバッガ（独自仕様言語デバッガ32に相当する。）のいずれかのデバッガを起動させるかのデバッグ選択ルーチンを実行する。

第7図は、このデバッグ選択ルーチンを示すフローチャートである。統合デバッガ35は、単一オブジェクトファイル中において順次実行されているステートメントがブレークポイントに到達すると、そのブレークポイントが設定されたス

ステートメントを表示する（ステップS201）。そして、そのステートメントがATLオブジェクトコード部に相当するならば（ステップS202：Yes）、ATLデバッガを起動し（ステップS206）、ATLデバッガからブレイクしたステートメントに含まれる変数等のデバッグ情報を取得する（ステップS207）。なお、ATLデバッガは、上記したブレイクポイント設定時に、統合デバッガ35によってATLオブジェクトコード部のブレイクポイント設定情報を取得している。

統合デバッガ35は、ATLデバッガからデバッグ情報を取得すると、ブレイク時に有効となっている指定変数（シンボル）を表示する（ステップS205）。

第8図は、統合デバッガ35の実行画面の例を示す図であり、特にATL記述部内でブレイクした状態を示している。第8図において、統合デバッガ35は、実行ウィンドウ50内に、ウィンドウ表示に標準として付加されるタイトルバーやメニューバーに加え、ブレイクポイント設定領域51、ソース表示領域52、シンボル表示領域53を有する。第8図では、C+ATLソースプログラム中のATL記述部内でブレイクした状態として、ブレイクポイントが設定された行番号5のステートメントがソース表示領域52内に表示されるとともに、シンボル表示領域53内に、そのステートメントで使用されている変数と変数に格納された値が表示されている。

一方、統合デバッガ35は、ブレイクしたステートメントがCオブジェクトコード部に相当するならば（ステップS202：No）、Cデバッガを起動し（ステップS203）、Cデバッガからブレイクしたステートメントに含まれる変数等のデバッグ情報を取得する（ステップS204）。なお、Cデバッガは、上記したブレイクポイント設定時に、統合デバッガ35によってCオブジェクトコード部のブレイクポイント設定情報を取得している。

統合デバッガ35は、Cデバッガからデバッグ情報を取得すると、ブレイク時に有効となっている指定変数（シンボル）を表示する（ステップS205）。第9図は、統合デバッガ35の実行画面の例を示す図であり、特にC言語記述部内

でブレークした状態を示している。第9図に示す統合デバッガ35の実行ウィンドウ50は、第8図と同様な構成である。第9図では、C+ATLソースプログラム中のC言語記述部内でブレークした状態として、ブレークポイントが設定された行番号15のステートメントがソース表示領域52内に表示されるとともに、シンボル表示領域53内に、そのステートメントで使用されている変数と変数に格納された値が表示されている。

プログラム開発者は、この統合デバッガ35のウィンドウ画面上に表示された変数の値を確認することによって論理的なエラーを発見した場合、エディタ28を起動させ、C+ATLソースファイルを適宜修正して上記したステップS120～S150の手順を繰り返す。

以上に説明したとおり、実施の形態にかかる半導体試験システム、すなわち本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法によれば、独自仕様言語プログラムであるATLソース自体を汎用言語プログラムであるC言語ソース内に埋め込むことによって、従来別管理されていた双方のソースを一つのC+ATLソースファイルとして取り扱うことができるので、ファイル管理を楽にし、プログラムの開発効率を向上させることができる。

なお、本実施の形態では、本発明にかかるプログラム開発支援装置とプログラム実行装置をそれぞれワークステーションと半導体試験装置に適用した形態を例示したが、プログラム開発支援装置を汎用的なコンピュータシステムとし、プログラム実行装置をそのコンピュータシステムと通信可能な測定装置や制御装置に適用することができるのは言うまでもない。

以上に説明したように、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法によれば、独自仕様言語プログラム自体を汎用言語プログラム内に埋め込むことで、従来別管理されていた双方のソースプログラムを、ソースファイルのみでなく、コンパイルによって作成されるオブジェクトファイルの段階においても一つのファイルとして取り扱うことが

できるので、ファイル管理を楽にし、プログラムの開発効率を向上させることができるという効果を奏する。

産業上の利用可能性

- 5 以上のように、本発明にかかるプログラム開発支援装置、プログラム実行装置、コンパイル方法およびデバッグ方法は、高性能な電子機器のプログラム（ファームウェア）を効率的に開発し、かつそのプログラムの管理を容易にするのに有用であり、特に、半導体試験装置のプログラムの開発および管理に適している。

請 求 の 範 囲

1. 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置
5 上で実行可能なプログラムファイルを作成するためのプログラム開発支援装置において、

前記独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイル手段と、

10 前記異種言語混在ソースプログラム内の前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイル手段と、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出し、抽出した独自仕様言語ソースプログラムを指定して前記独自仕様言語コンパイル手段を実行させるとともに、前記異種言語混在ソースプログラムを指定
15 して前記汎用言語コンパイル手段を実行させ、得られた独自仕様言語オブジェクトコードと汎用言語オブジェクトコードを結合してオブジェクトファイルを生成する統合コンパイル手段と、

前記統合コンパイル手段によって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンク手段と、
20 を備えたことを特徴とするプログラム開発支援装置。

2. 前記統合コンパイル手段は、前記オブジェクトファイルに、前記独自仕様言語オブジェクトコードおよび／または前記汎用言語オブジェクトコードのコード位置情報を付加することを特徴とする請求の範囲第1項に記載のプログラム開発支援装置。
25

3. 前記プログラムファイルを前記プログラム実行装置に転送するプログラム

転送手段を備えたことを特徴とする請求の範囲第1項に記載のプログラム開発支援装置。

4. 前記プログラム実行装置に対して該プログラム実行装置に転送されたプログラムファイルを実行させる指示を与えるプログラム実行装置コントロール手段を備えたことを特徴とする請求の範囲第3項に記載のプログラム開発支援装置。

5. 前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定手段と、

前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動することを特徴とする請求の範囲第1項に記載のプログラム開発支援装置。

6. 前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報を共通するウィンドウ画面に表示することを特徴とする請求の範囲第5項に記載のプログラム開発支援装置。

7. 前記汎用言語はC言語であり、前記独自仕様言語ソースプログラムは、前記汎用言語ソースプログラムのプリプロセッサコマンドによって該汎用言語ソースプログラム内に記述されたことを特徴とする請求の範囲第1項に記載のプログラム開発支援装置。

8. 前記プリプロセッサコマンドは、`#pragma`であることを特徴とする請求の範囲第7項に記載のプログラム開発支援装置。

9. 前記プログラム実行装置は、半導体試験装置であることを特徴とする請求項1に記載のプログラム開発支援装置。

5 10. 汎用言語ソースプログラムのオブジェクトコードと独自仕様言語ソースプログラムのオブジェクトコードが混在したプログラムファイルを実行するプログラム実行装置において、

前記プログラムファイルの実行開始時に、前記汎用言語ソースプログラムのオブジェクトコードと前記独自仕様言語ソースプログラムのオブジェクトコードを
10 メモリにロードすることを特徴とするプログラム実行装置。

11. 前記プログラム実行装置は半導体試験装置であることを特徴とする請求の範囲第10項に記載のプログラム実行装置。

15 12. 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装置上で実行可能なプログラムファイルを作成するためのコンパイル方法において、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを抽出する独自仕様言語ソースプログラム抽出ステップと、

20 抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブジェクトコードを生成する独自仕様言語コンパイルステップと、

前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップと、

25 前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップと、

前記オブジェクトファイル生成ステップによって生成された少なくとも一つの

オブジェクトファイルから前記プログラムファイルを生成するリンクステップと、
を含んだことを特徴とするコンパイル方法。

1 3. 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラム
5 が記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム
実行装置上で実行可能なプログラムファイルをデバッグするためのデバッグ
方法において、

前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定する
ブレークポイント設定ステップと、

10 前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイル
を停止させるとともに、停止したプログラムファイルの前記ステートメント
が前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停
止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプロ
グラムに属する場合に独自仕様言語デバッガを起動するデバッグ起動ステップと、

15 前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情
報を共通するウィンドウ画面に表示するデバッグ情報表示ステップと、
を含んだことを特徴とするデバッグ方法。

1 4. 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラム
20 が記述された構成の異種言語混在ソースプログラムから所定のプログラム実行装
置上で実行可能なプログラムファイルを作成するためのステップをコンピュータ
に実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体におい
て、

前記異種言語混在ソースプログラムから前記独自仕様言語ソースプログラムを
25 抽出する独自仕様言語ソースプログラム抽出ステップと、

抽出された独自仕様言語ソースプログラムをコンパイルして独自仕様言語オブ
ジェクトコードを生成する独自仕様言語コンパイルステップと、

前記異種言語混在ソースプログラムのうちの前記汎用言語ソースプログラム記述部分をコンパイルして汎用言語オブジェクトコードを生成する汎用言語コンパイルステップと、

- 5 前記独自仕様言語オブジェクトコードと前記汎用言語オブジェクトコードを結合してオブジェクトファイルを生成するオブジェクトファイル生成ステップと、

前記オブジェクトファイル生成ステップによって生成された少なくとも一つのオブジェクトファイルから前記プログラムファイルを生成するリンクステップと、を含むことを特徴とするコンピュータ読み取り可能な記録媒体。

- 10 15. 汎用言語ソースプログラムの所定領域に独自仕様言語ソースプログラムが記述された構成の異種言語混在ソースプログラムから作成された所定のプログラム実行装置上で実行可能なプログラムファイルをデバッグするためのステップをコンピュータに実行させるプログラムを記録したコンピュータ読み取り可能な記録媒体において、

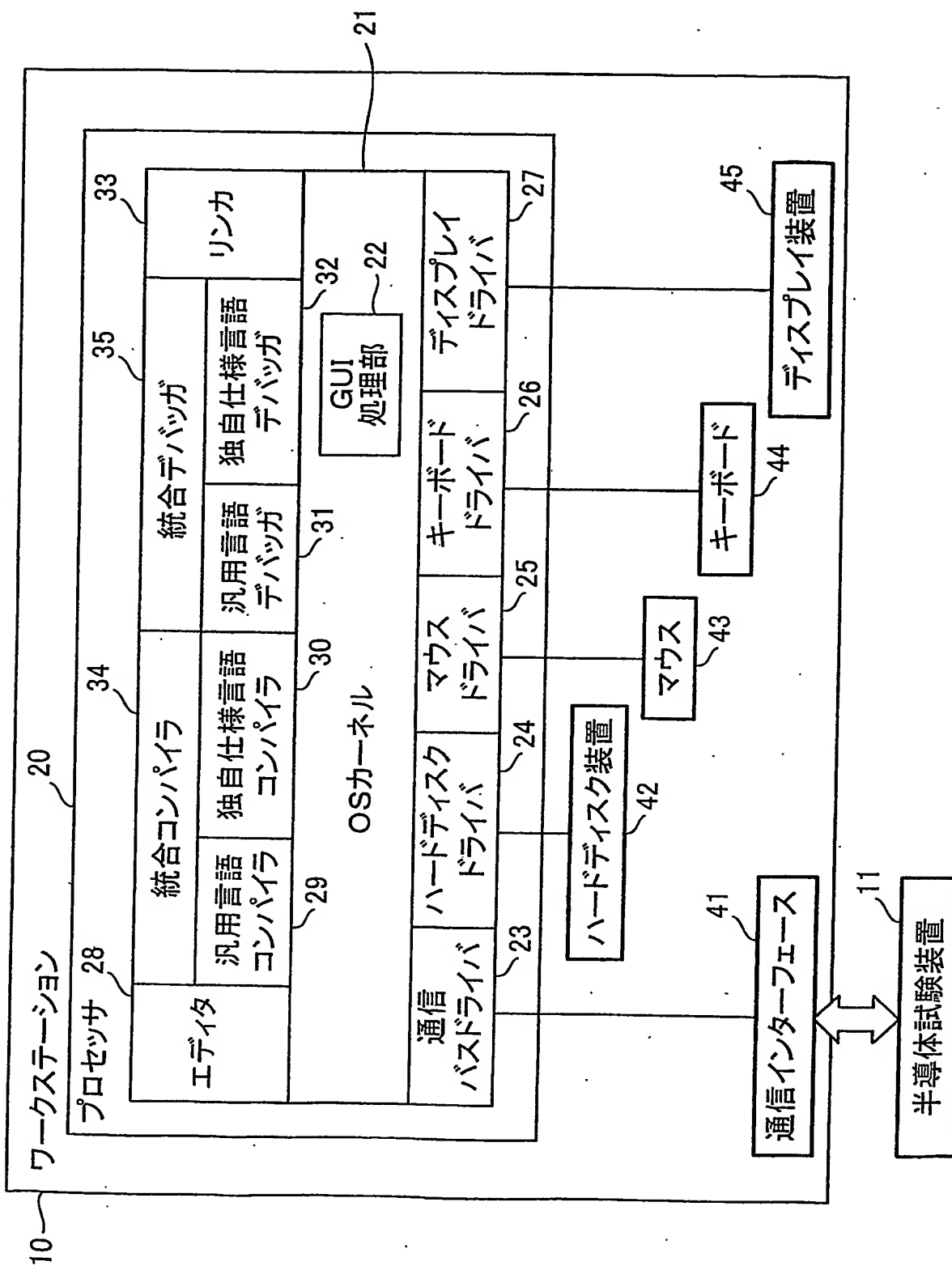
- 15 前記異種言語混在ソースプログラム内のステートメントにブレークポイントを設定するブレークポイント設定ステップと、

- 前記プログラムファイルの実行時に前記ブレークポイントで該プログラムファイルを停止させるとともに、停止したプログラムファイルの前記ステートメントが前記汎用言語ソースプログラムに属する場合に汎用言語デバッガを起動し、停止したプログラムファイルの前記ステートメントが前記独自仕様言語ソースプログラムに属する場合に独自仕様言語デバッガを起動するデバッガ起動ステップと、

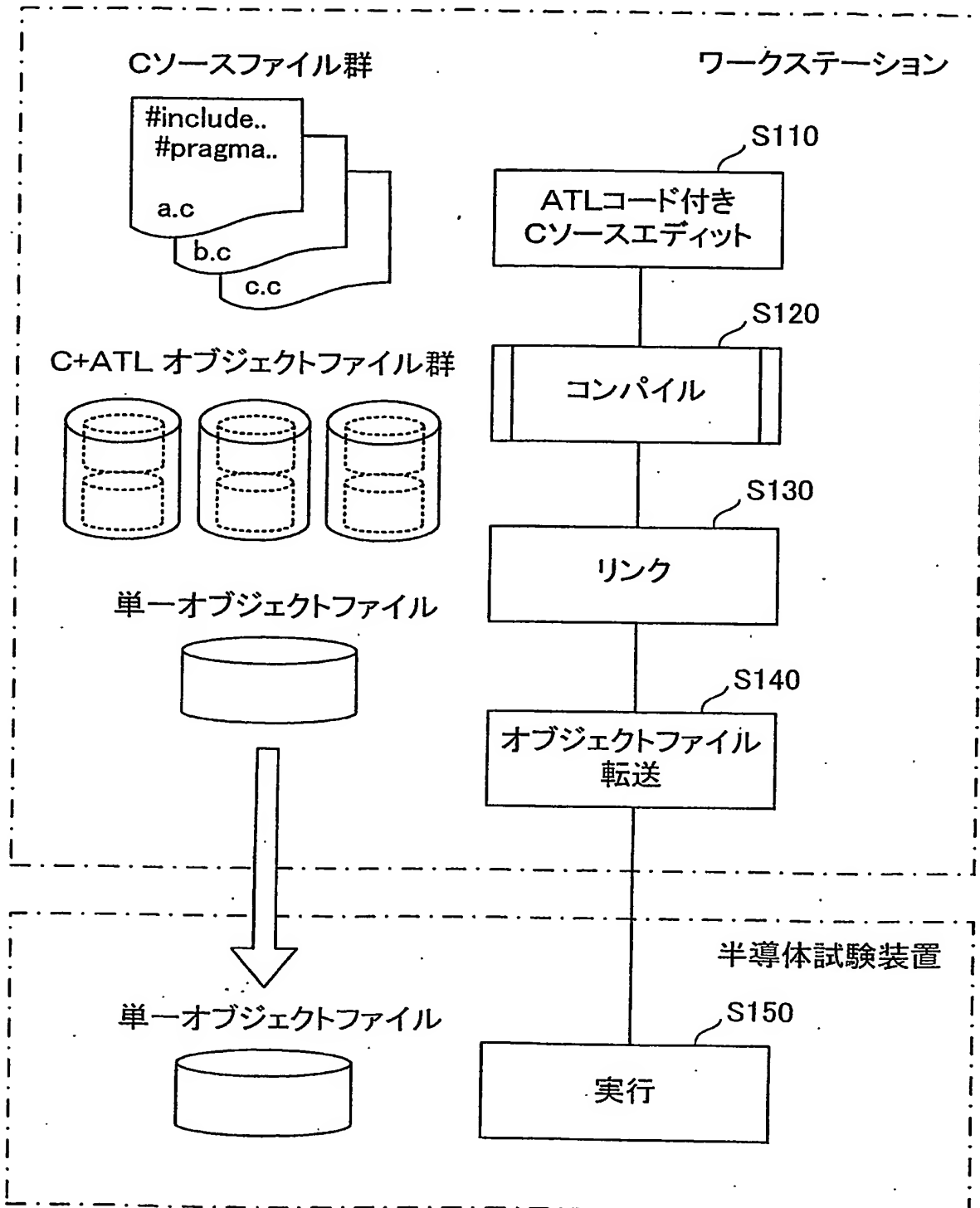
前記汎用言語デバッガと前記独自仕様言語デバッガとから得られたデバッグ情報を共通するウィンドウ画面に表示するデバッグ情報表示ステップと、

を含んだことを特徴とするコンピュータ読み取り可能な記録媒体。

第1図



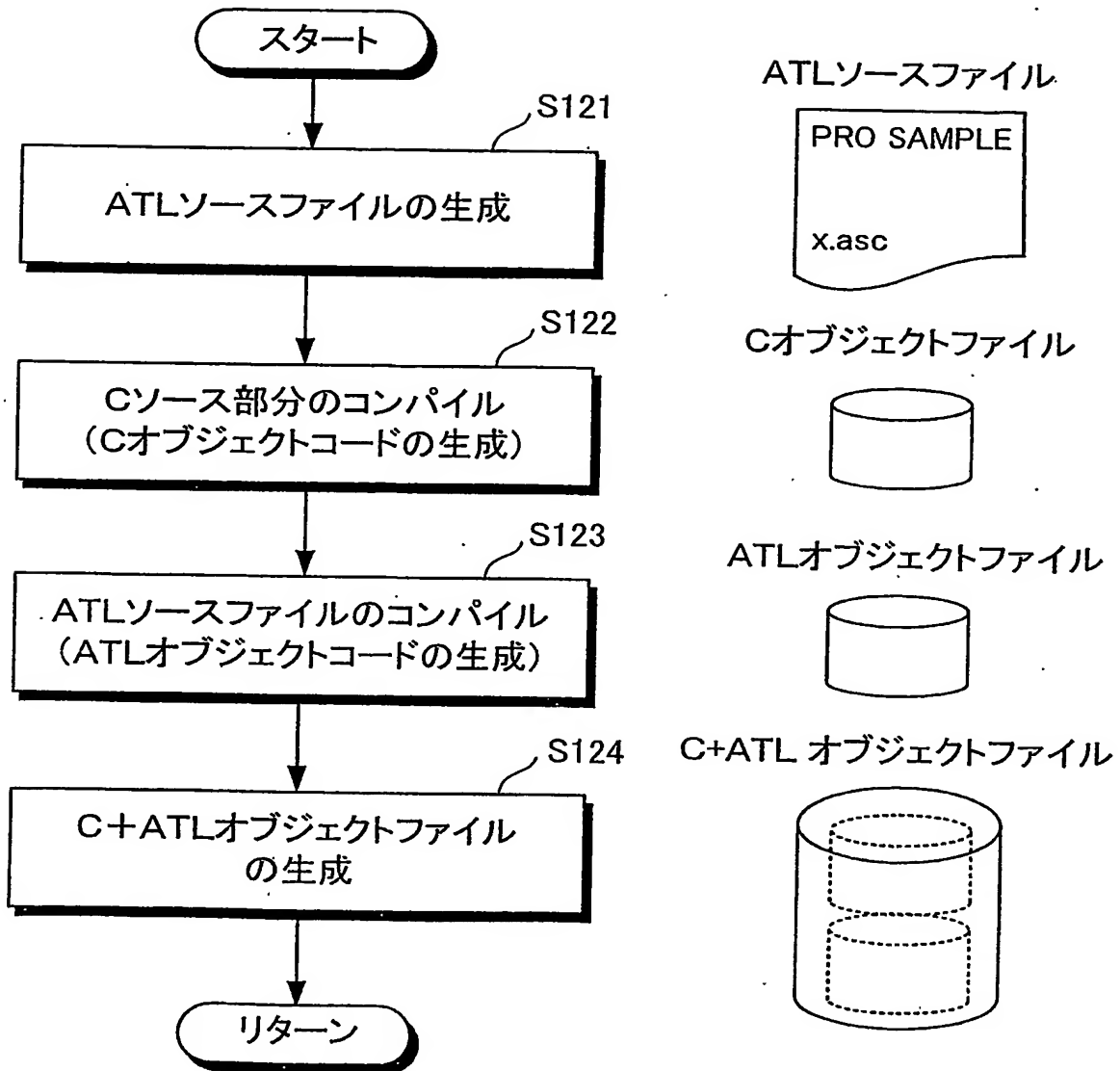
第2図



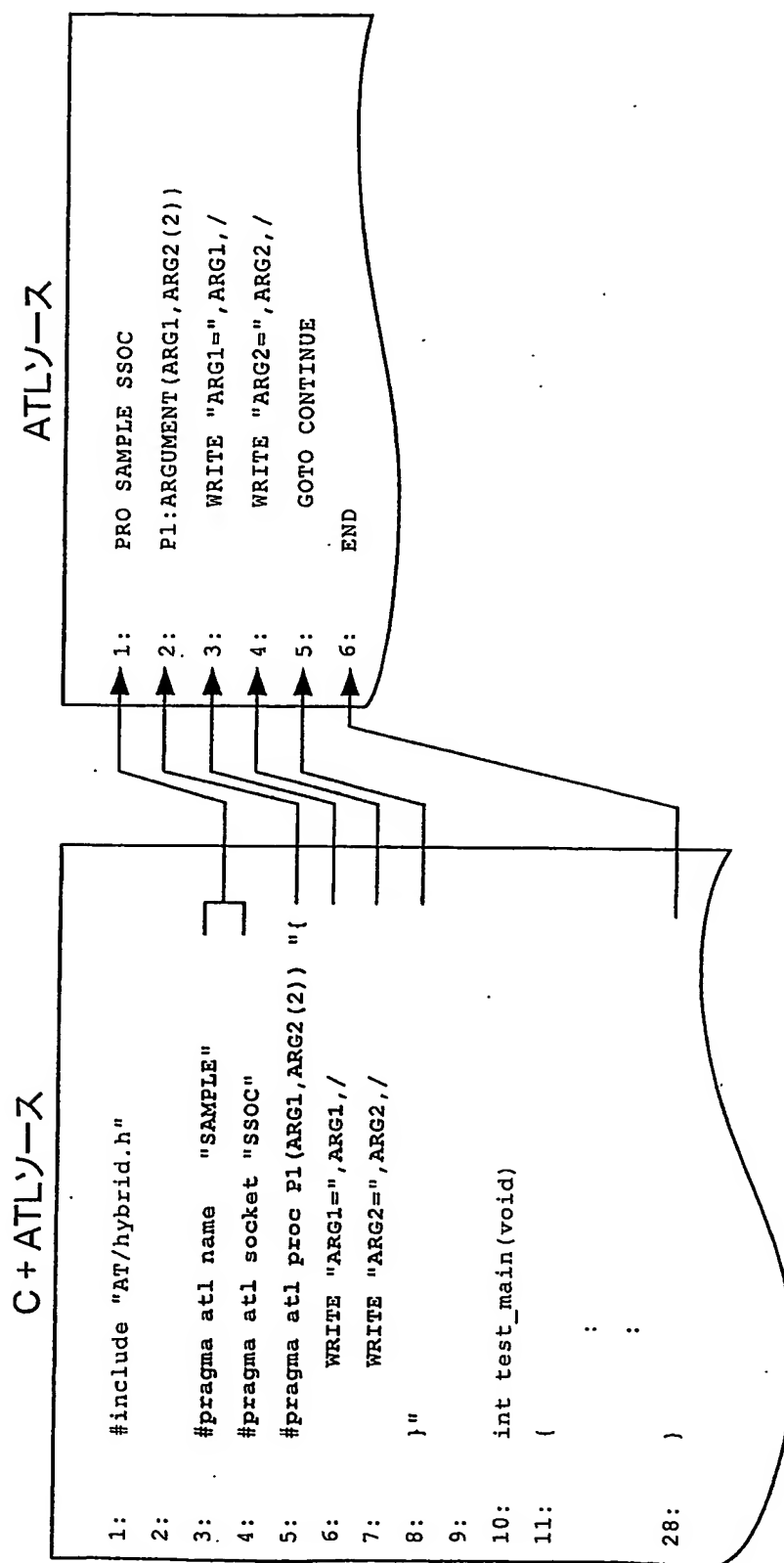
第3図

```
1:  #include "AT/hybrid.h"
2:
3:  #pragma atl name      "SAMPLE"
4:  #pragma atl socket    "SSOC"
5:  #pragma atl proc P1(ARG1,ARG2(2)) "{
6:      WRITE "ARG1=",ARG1,/
7:      WRITE "ARG2=",ARG2,/
8:  }"
9:
10: int test_main(void)
11: {
12:     int idata;
13:     struct atreal rdata[2];
14:
15:     if(ATL_init2(EM_ATL) != 0){
16:         AT_perror("ATL_init2()");
17:         exit(1);
18:     }
19:
20:     idata = 1;
21:     AT_strtorm(rdata, 2, "0.5V", "1.2V");
22:     if(ATL_proc(EM_ATL, "P1", INTE(&idata), DREAL(rdata, 2)) != 0){
23:         AT_perror("P2");
24:         exit(1);
25:     }
26:
27:     return(0);
28: }
```

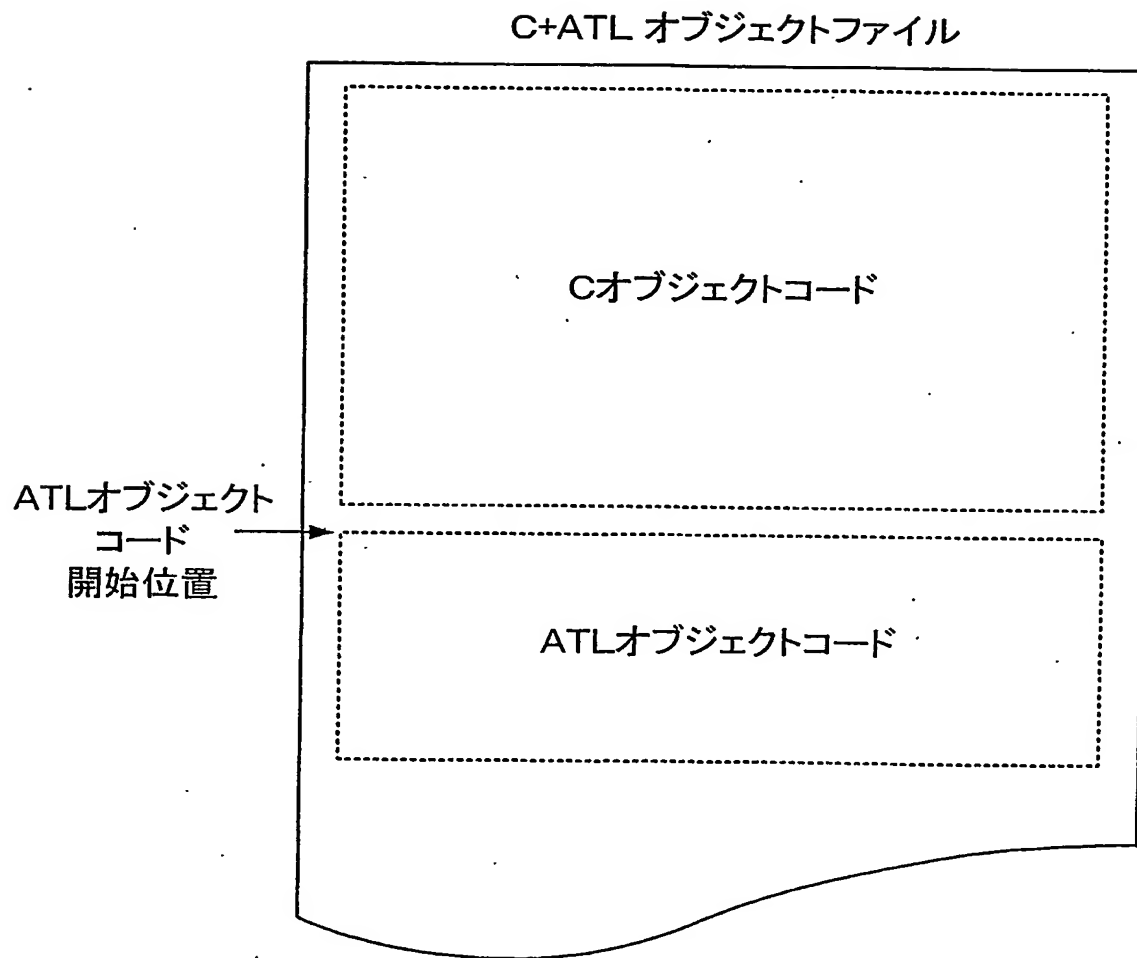

第4図



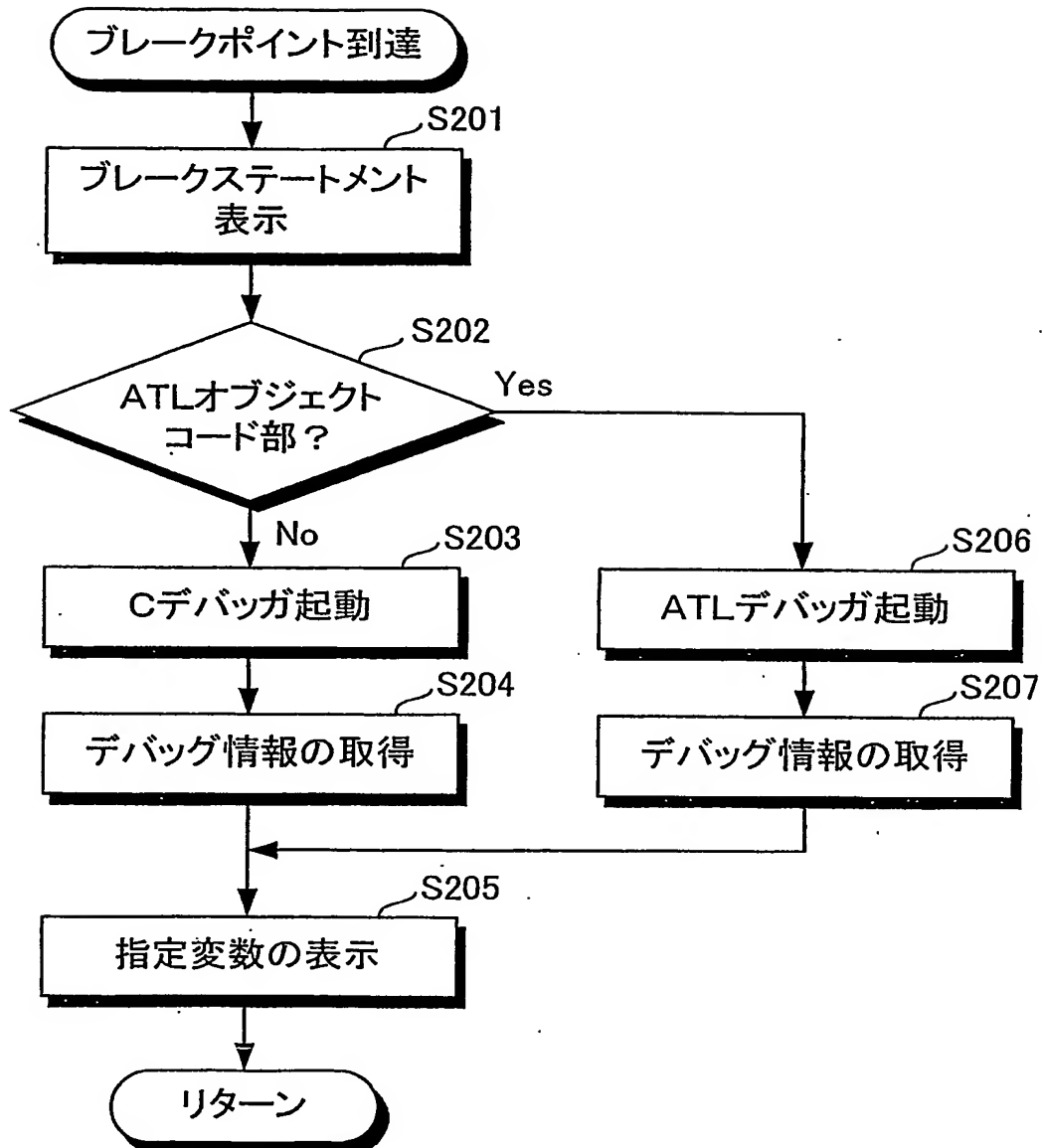
第5図



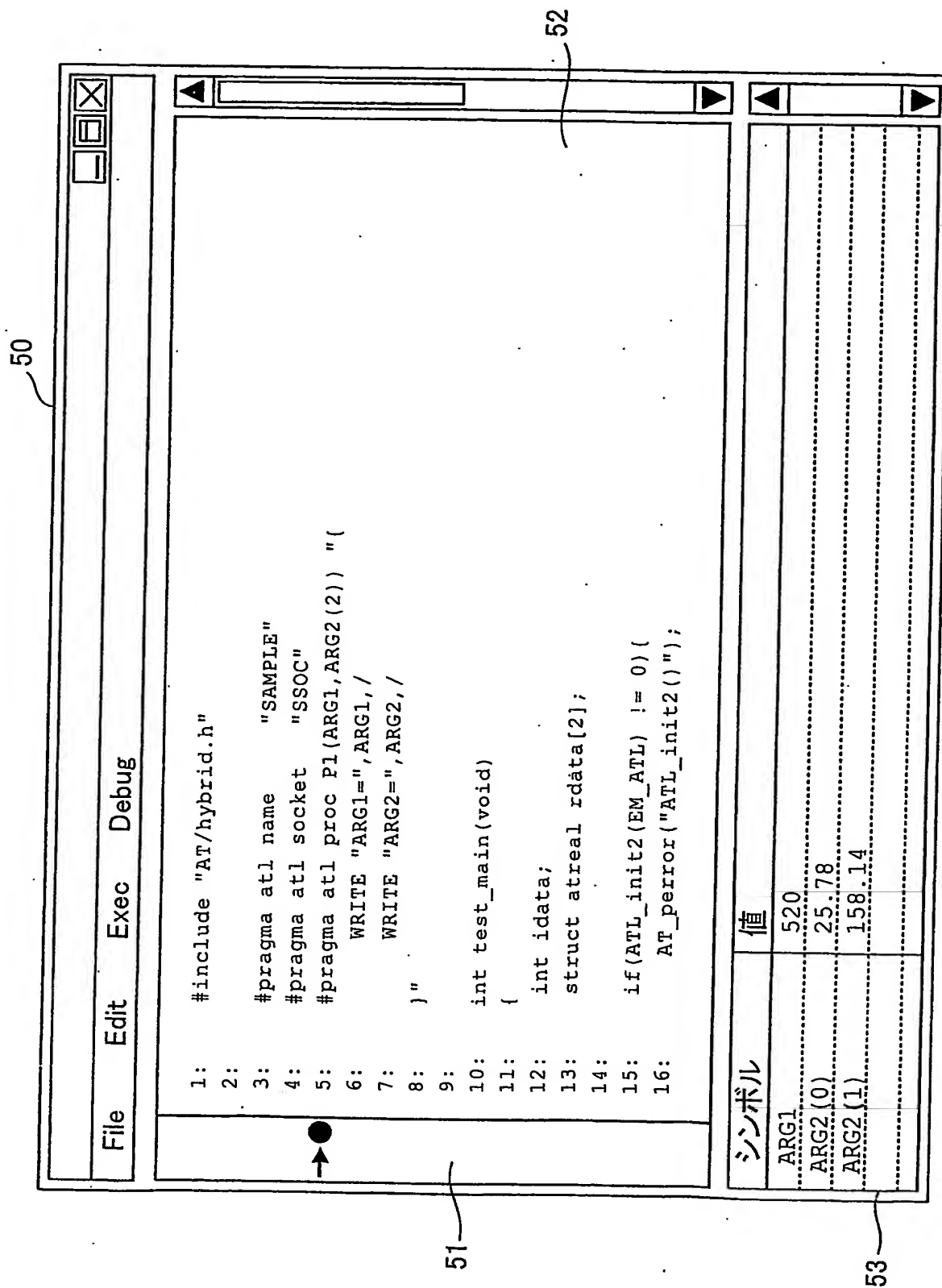
第 6 図



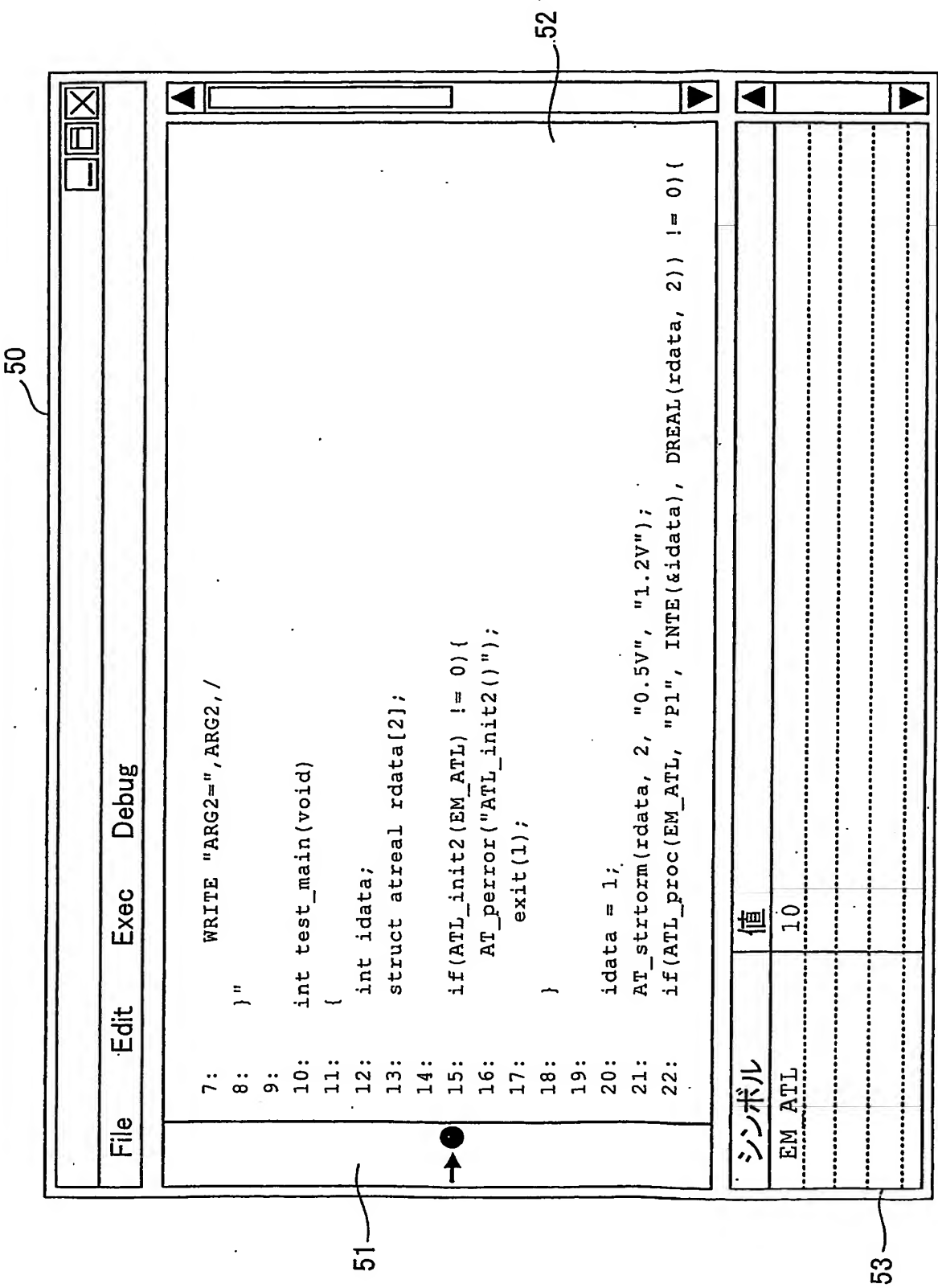
第7図



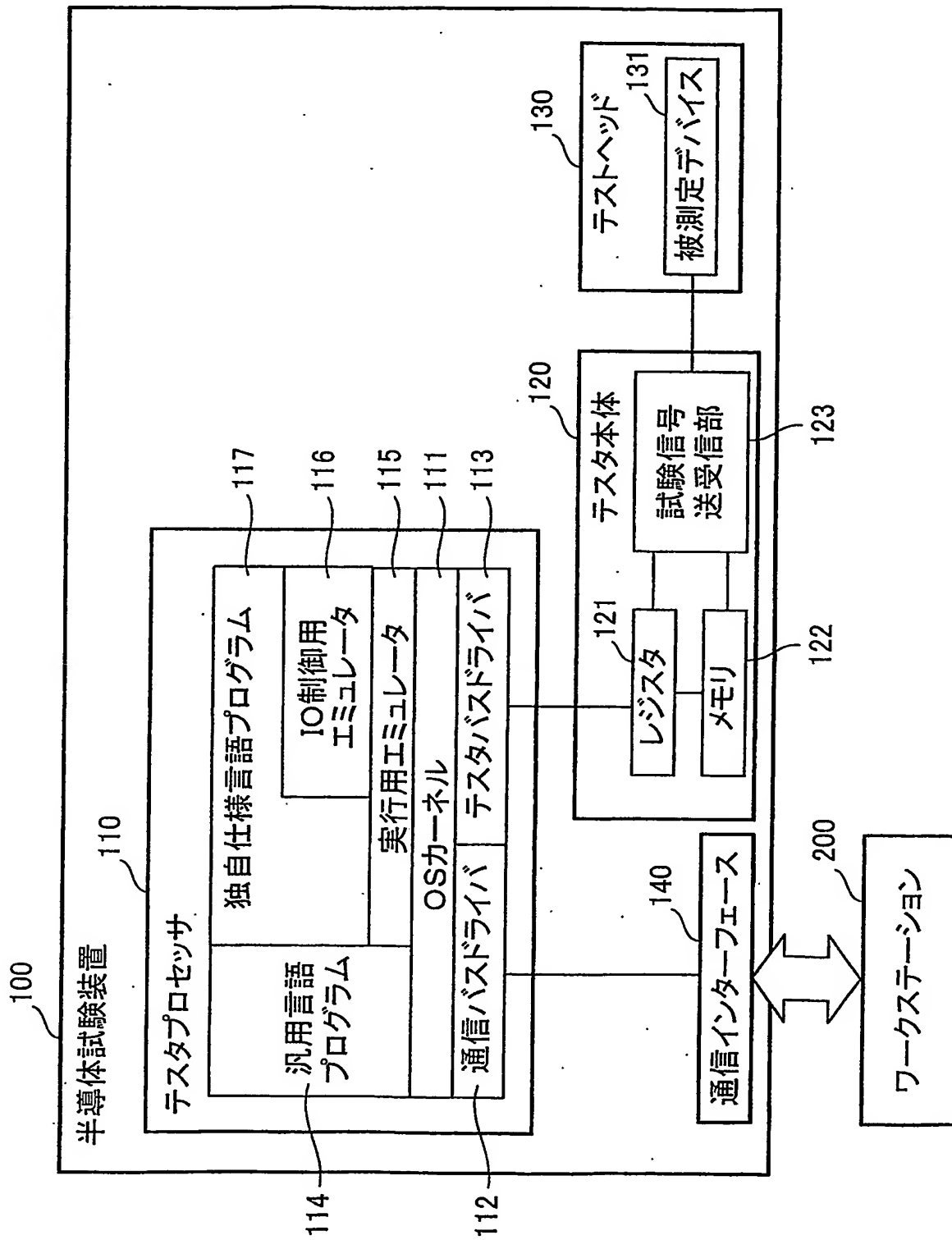
第 8 図



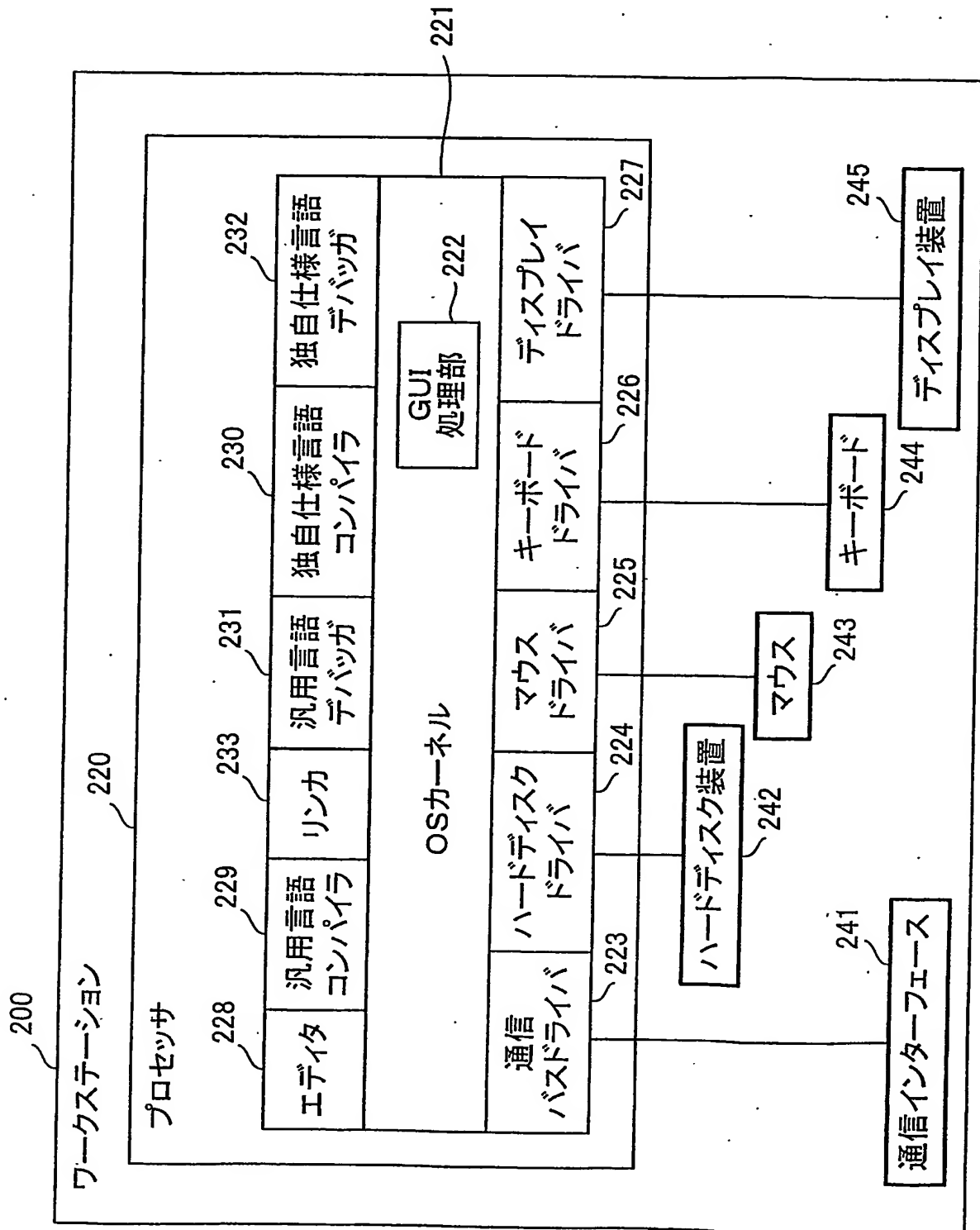
第 9 図



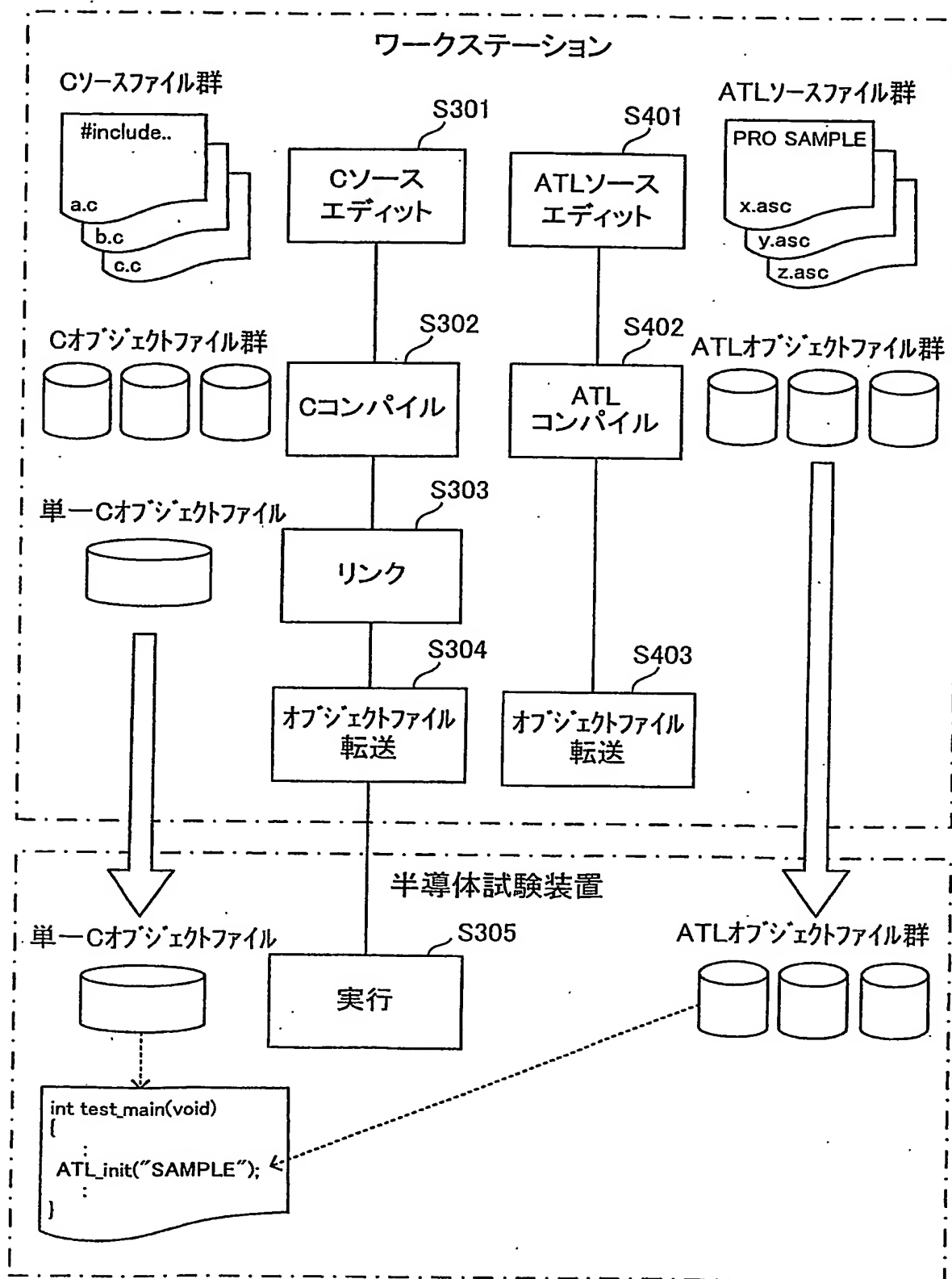
第10図



第 1 1 図



第12図



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP03/13302

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl⁷ G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl⁷ G06F9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1922-1996	Toroku Jitsuyo Shinan Koho	1994-2003
Kokai Jitsuyo Shinan Koho	1971-2003	Jitsuyo Shinan Toroku Koho	1996-2003

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 11-212807 A (Hitachi, Ltd.), 06 August, 1999 (06.08.99), Full text; Figs. 1 to 4 (Family: none)	1-15
A	JP 7-319729 A (Hitachi, Ltd.), 08 December, 1995 (08.12.95), Full text; Figs. 1 to 5 (Family: none)	1-15
A	JP 2002-268896 A (Hitachi, Ltd.), 20 September, 2002 (20.09.02), Full text; Figs. 1 to 6 (Family: none)	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance
 "E" earlier document but published on or after the international filing date
 "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
 "O" document referring to an oral disclosure, use, exhibition or other means
 "P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
 "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
 "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
 "&" document member of the same patent family

Date of the actual completion of the international search
04 December, 2003 (04.12.03)

Date of mailing of the international search report
16 December, 2003 (16.12.03)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP03/13302

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2-146630 A (Fujitsu Maikon Systems Kabushiki Kaisha), 05 June, 1990 (05.06.90), Full text; Figs. 1 to 11 (Family: none)	1-15
A	JP 11-110256 A (Toshiba Corp.), 23 April, 1999 (23.04.99), Full text; Figs. 1 to 11 (Family: none)	1-15

A. 発明の属する分野の分類 (国際特許分類 (IPC))

Int. Cl. G06F9/44

B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (IPC))

Int. Cl. G06F9/44

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1922-1996年
 日本国公開実用新案公報 1971-2003年
 日本国登録実用新案公報 1994-2003年
 日本国実用新案登録公報 1996-2003年

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	JP 11-212807 A (株式会社日立製作所) 1999. 08. 06, 全文, 第1-4図 (ファミリーなし)	1-15
A	JP 7-319729 A (株式会社日立製作所) 1995. 12. 08, 全文, 第1-5図 (ファミリーなし)	1-15
A	JP 2002-268896 A (株式会社日立製作所) 2002. 09. 20, 全文, 第1-6図 (ファミリーなし)	1-15

☒ C欄の続きにも文献が列挙されている。☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」 特に関連のある文献ではなく、一般的技術水準を示すもの
 「E」 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
 「L」 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)
 「O」 口頭による開示、使用、展示等に言及する文献
 「P」 国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」 国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「X」 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Y」 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「&」 同一パテントファミリー文献

国際調査を完了した日

04. 12. 03

国際調査報告の発送日

16.12.03

国際調査機関の名称及びあて先

日本国特許庁 (ISA/JP)
 郵便番号100-8915
 東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

後藤 和茂

5B

9463

電話番号 03-3581-1101 内線 6907

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	J P 2-146630 A (富士通マイコンシステムズ株式会社) 1990. 06. 05, 全文, 第1-11図 (ファミリーなし)	1-15
A	J P 11-110256 A (株式会社東芝) 1999. 04. 23, 全文, 第1-11図 (ファミリーなし)	1-15